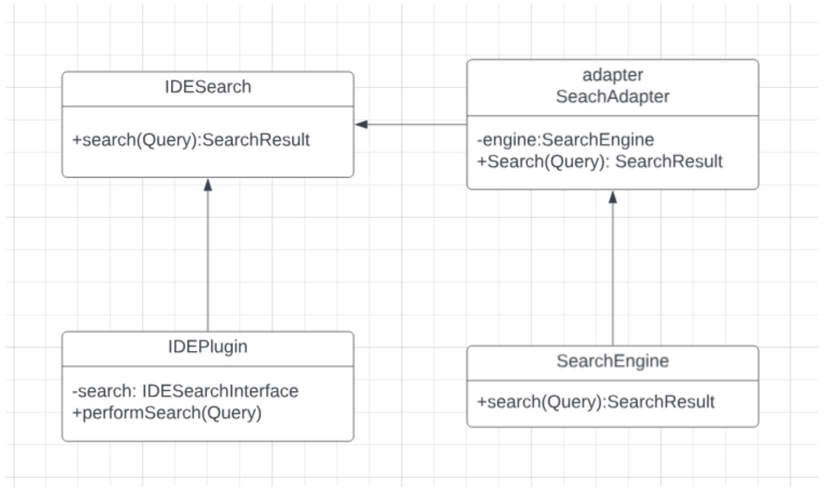
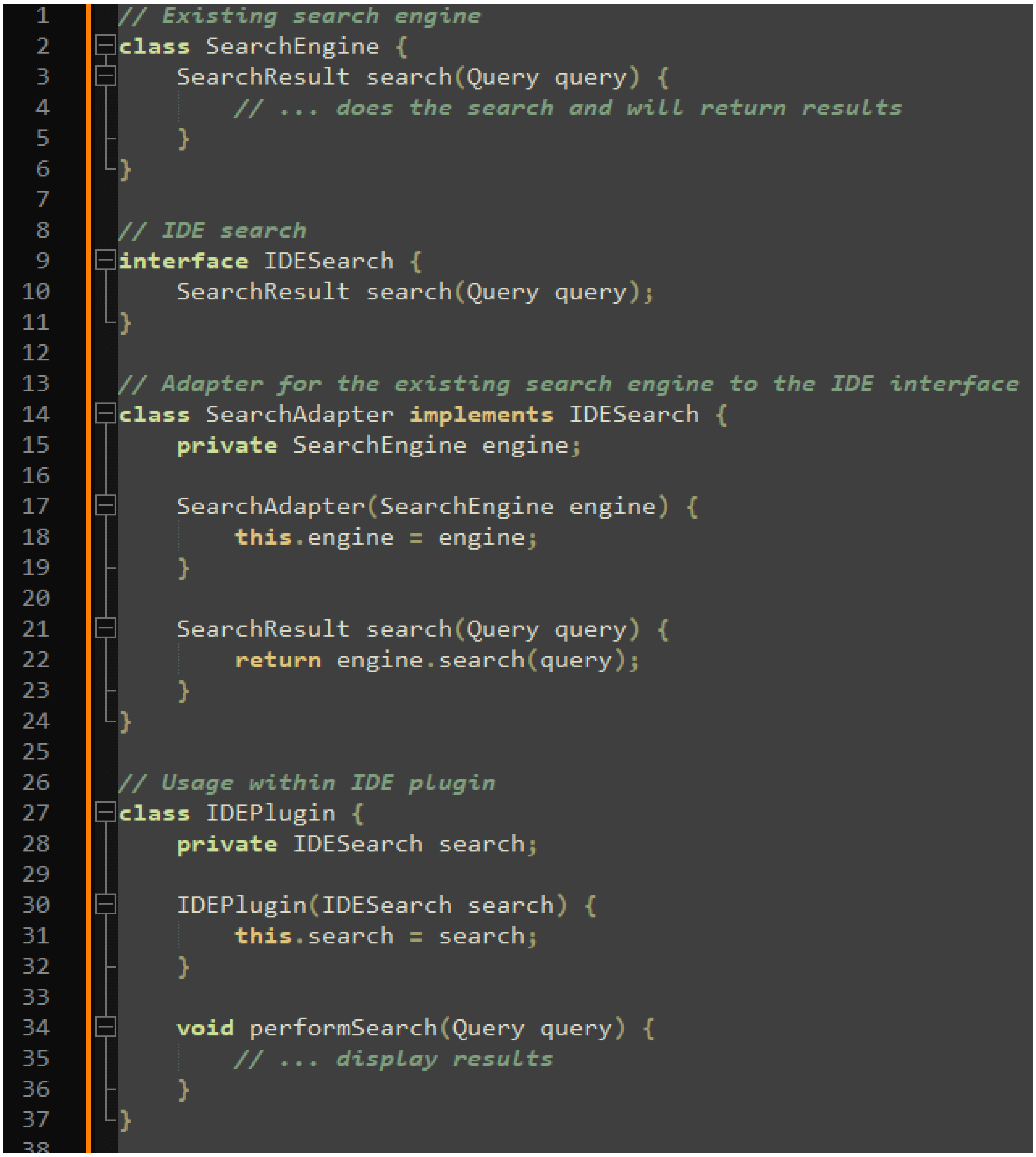
High Level Design

The architectural pattern that we would use to structure the system would be using the Model-View-Controller (MVC) architecture. MVC follows a pattern that starts with the user using the software, then the controller stage manipulates and handles inputs and then sends it to the model, which then updates and sends outputs to the view, which handles the UI and then displays the information back to the user. Our project deals with creating an AI plugin inside of an IDE that deals with helping a user conveniently find solutions to their bugs and also serves as a coding search engine. In this situation, the user would use the plugin, and the Controller would handle the user's inputs, process requests, manage things such as user interactions, and deal with functionality such as triggering code analysis. The Model represents the AI aspects that are responsible for code analysis and solutions and this stage follows the Controller stage, where the actual code is being processed. Then, the results from the Model stage will be passed to the View stage, which presents the data to the user and deals with the UI of the program. This would be the part that interacts with the IDE's UI and would display the information and feedback to the user in an aesthetically pleasing way. Finally, the user will see all of the information that is presented, and this cycle will continue. Using MVC as the architectural pattern is extremely efficient for our project and makes the design flow easy to break down and implement. The codebase would be more maintainable and would allow for separation within development as well as flexibility.

Low-Level Design

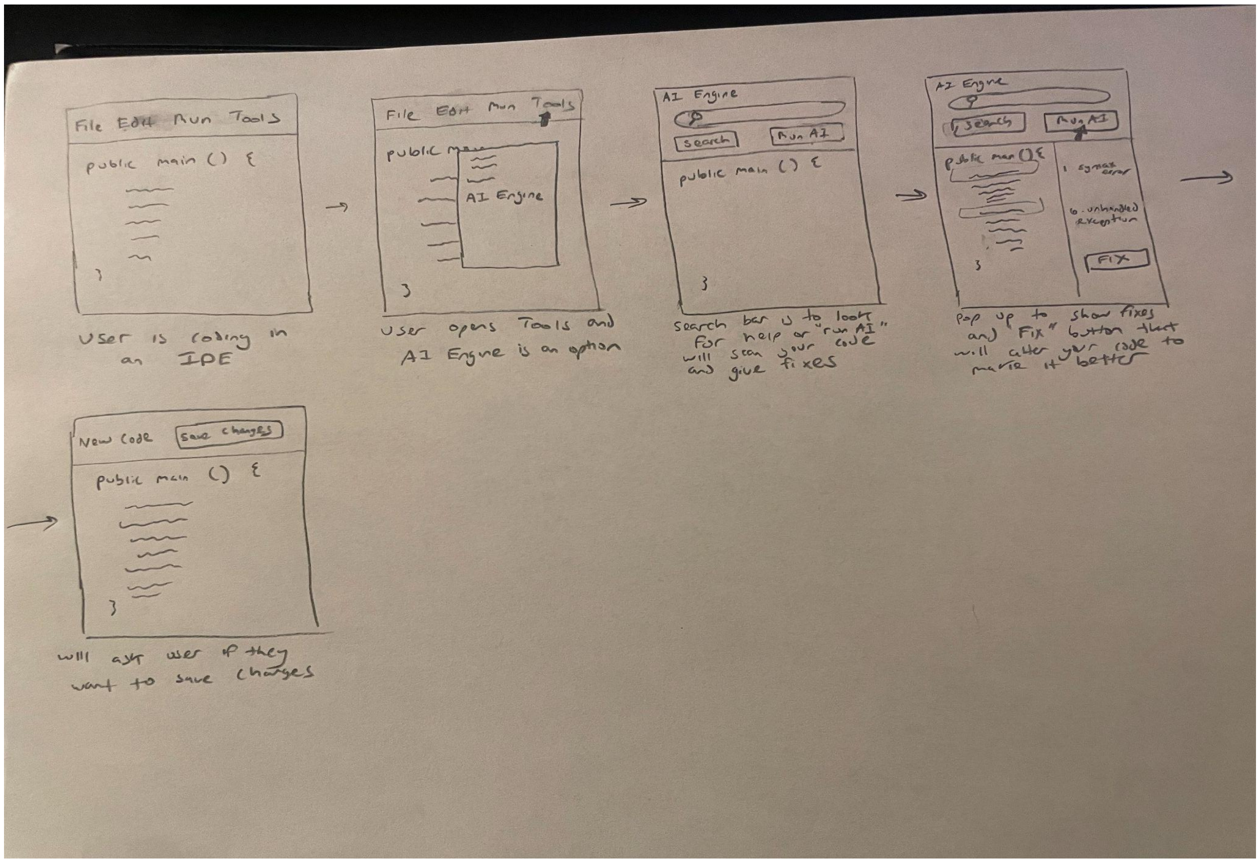
For the low-level design aspect, the best design pattern family would be 'structural' because this type of design pattern deals with classes and objects and will allow us to add new functionality or add more structures to our project if we need them in the future. The 'adapter' would be the most useful design pattern for us as this deals with APIs and based on what we are going to do, we would be using various APIs for different sorts of functionality within the IDE.


IDESearch
+search(Query):SearchResult
adapter
SeachAdapter
-engine:SearchEngine
+Search(Query): SearchResult
IDEPlugin
-search: IDESearchInterface
+performSearch(Query)
SearchEngine
+search(Query):SearchResult



```
// Existing search engine
class SearchEngine {
    SearchResult search(Query query) {
        // ... does the search and will return results
    }
}

// IDE search
interface IDESearch {
    SearchResult search(Query query);
}

// Adapter for the existing search engine to the IDE interface
class SearchAdapter implements IDESearch {
    private SearchEngine engine;

    SearchAdapter(SearchEngine engine) {
        this.engine = engine;
    }

    SearchResult search(Query query) {
        return engine.search(query);
    }
}

// Usage within IDE plugin
class IDEPlugin {
    private IDESearch search;

    IDEPlugin(IDESearch search) {
        this.search = search;
    }

    void performSearch(Query query) {
        // ... display results
    }
}
```

Design Sketch



While this program will have multiple purposes, a main one includes a user using the software to have the AI change their code automatically so it is more efficient. In this storyboard, the user is seen coding in their IDE. Then they open the tools section that will have an option for our IDE software and the user selects it. The dropdown for the software will have options such as running the AI to provide code fixes as well as a search bar that you can look up information such as how to write a for loop or how to make a class, etc. In this case, the user runs the AI for code fixes and a sidebar shows up with how to fix code as well as an automatic code fix button that the user presses. After pressing, the code has been changed and the user can choose to save changes or delete what the AI did.

Process Deliverable (Scrum)

Date: 11/6/2023
Time: 5pm
Location: Torg

Team Members Present:
Jae

Braden
Achintya
Mohammad

Agenda:

1. Update team members
2. Talk about next steps
3. Talk about any roadblocks

Meetings Notes:

Jae:

- Tasks Completed:
    - PM2
        - Requirement Analysis Use Case
    - PM3
        - High Level Design
- Tasks To Do:
    - PM3
        - Design Sketch

Mohammad:

- Tasks Competed:
    - Documented Scrum Meeting
    - PM2
        - Project Scrum Meeting Deliverable
        - Requirements Analysis Use Case
- Tasks To Do:
    - PM3
        - Scrum Meeting Deliverable
        - Schedule Future Scrum Meetings

Braden:

- Tasks Completed:
    - PM2
        - Requirements Analysis Use Case
    - PM3

        - Created GitHub Repository
- Tasks To Do:
    - PM3
        - Project Check-In Survey

Achintya:

- Tasks Completed:
    - PM2
        - Requirements Analysis Use Case
- Tasks To Do:
    - PM3
        - Low-Level Design